# Chombo-Crunch and VisIt for carbon sequestration and in-transit data analysis using burst buffers

**Andrey Ovsyannikov (NERSC),**
Melissa Romanus (Rutgers U.),
Brian Van Straalen (LBL), Gunther Weber (LBL),
David Trebotich (LBL)

April 18-21, 2016
Glendale, AZ

# Outline

- Motivation
- Conventional I/O and alternatives
- Burst Buffer architecture
- Proposed approach: asynchronous workflow
- Chombo-Crunch example
- Results
- Conclusions

# Motivation

**Emerging exascale systems one has to deal with:**

- **Growing amount of data at an unprecedented rate**
- **Insufficient bandwidth of persistent storage media. Growing gap between computation and I/O rates**
- **Scientific workflows are getting more complex. Exchange of data between different workflow components is getting challenging**

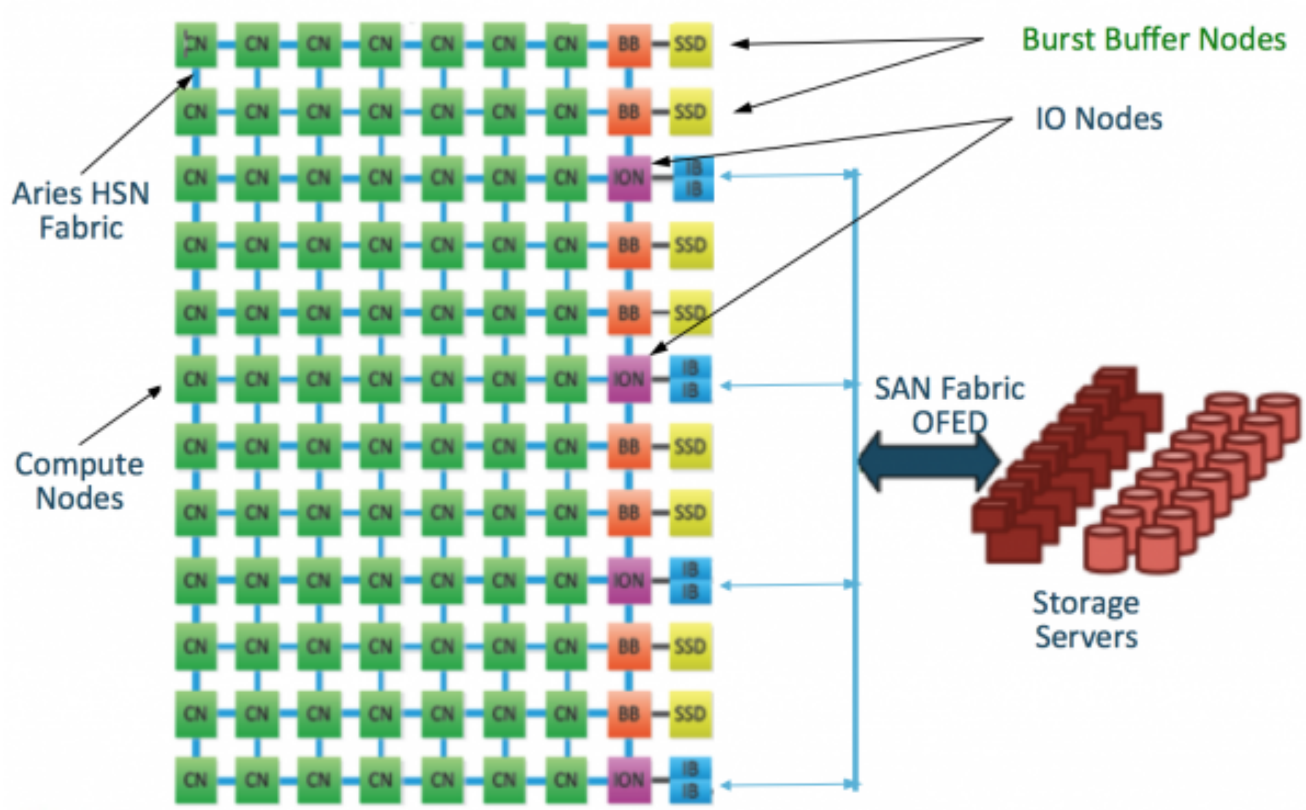
**Need of alternatives to conventional post-processing approach**

# Different data analysis methods

| | *In situ* | **In Transit** | **Post Processing** |
|---|---|---|---|
| **Analysis Execution Location** | Within Simulation | Burst Buffer | Separate Application |
| **Data Location** | Within Simulation Memory Space | Within Burst Buffer Flash Memory | On Parallel File System |
| **Data Reduction Possible?** | YES: Can limit output to only analysis products. | YES: Can limit data saved to disk to only analysis products. | NO: All data saved to disk for future use. |
| **Interactivity** | NO: Analysis actions must be pre-scripted to run within simulation. | LIMITED: Data is not permanently resident in flash and can be removed to disk. | YES: User has full control on what to load and when to load data from disk. |
| **Analysis Routines Expected** | Fast running analysis operations, statistical routines, image rendering. | Longer running analysis operations bounded by the time until drain to file system. Statistics over simulation time. | All possible analysis and visualization routines including interactive exploration of the rendered dataset. |

Comparison of data analysis execution methods (Prabhat & Koziol, 2015)

# Burst Buffer architecture

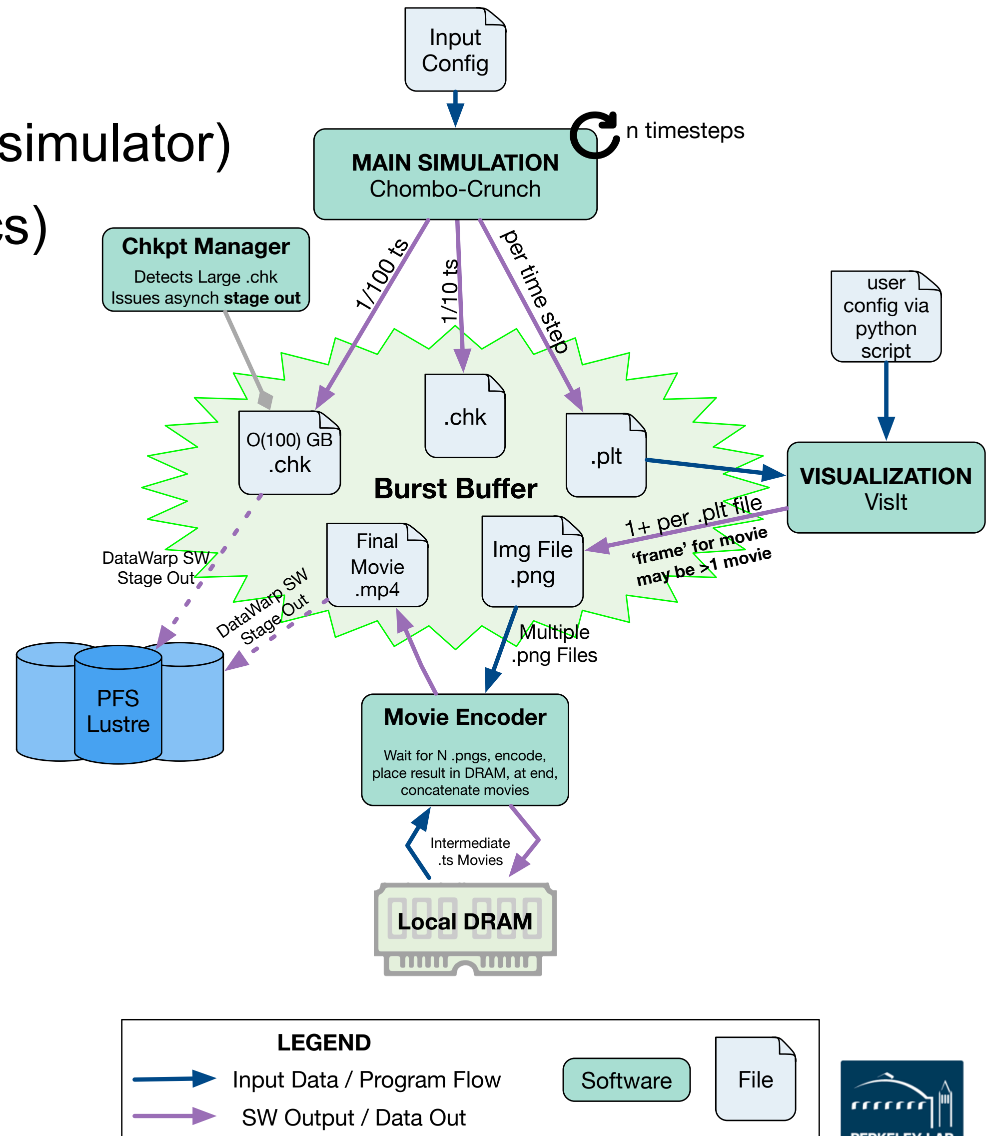



- Current configuration: 850TB on 144 BB nodes (288 SSDs)
- >1.5 PB total coming with Cori Phase 2

# Proposed in-transit workflow

Workflow components:

- ❑ **Chombo-Crunch** (subsurface simulator)
- ❑ **VisIt** (visualization and analytics)
- ❑ **Encoder**
- ❑ **Checkpoint manager**



# Slurm implementation

Allocate BB capacity

Stage in restart file

Run each component

Stage output file to PFS

```
#!/bin/bash
#SBATCH --nodes=1040
#SBATCH --job-name=shale
#DW jobdw capacity=200TiB access_mode=striped type=scratch
#DW stage_in type=file source=/pfs/restart.hdf5 destination
     =$DW_JOB_STRIPED/restart.hdf5
### Load required modules
module load visit
ScratchDir="$SLURM_SUBMIT_DIR/_output.$SLURM_JOBID"
BurstBufferDir="${DW_JOB_STRIPED}"
mkdir $ScratchDir
stripe_large $ScratchDir
NumTimeSteps=2000
EncoderInt=120
RestartFileName="restart.hdf5"
ProgName="chombocrunch3d.Linux.64.CC.ftn.OPTHIGH.MPI.PETSC.
ex"
ProgArgs=chombocrunch.inputs
ProgArgs="$ProgArgs check_file=${BurstBufferDir}check
     plot_file=${BurstBufferDir}plot pfs_path_to_checkpoint=
     ${ScratchDir}/check restart_file=${BurstBufferDir}${
     RestartFileName} max_step=$NumTimeSteps"
### Launch Chombo-Crunch
srun -N 1024 -n 32768 $ProgName $ProgArgs > log 2>&1 &
### Launch VisIt
visit -l srun -nn 16 -np 512 -cli -nowin -s VisIt.py &
### Launch Encoder
./encoder.sh -pngpath $BurstBufferDir -endts $NumTimeSteps
     -i $EncoderInt &
wait
### Stage-out movie file from Burst Buffer
#DW stage_out type=file source=$DW_JOB_STRIPED/movie.mp4
     destination=/pfs/movie.mp4
```

# Chombo-Crunch

Simulates pore scale reactive transport processes associated with carbon sequestration

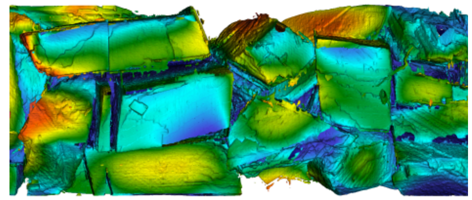
Applied to other subsurface science areas:

- Hydrofracturing
- Used fuel disposition (Hanford salt repository modeling)

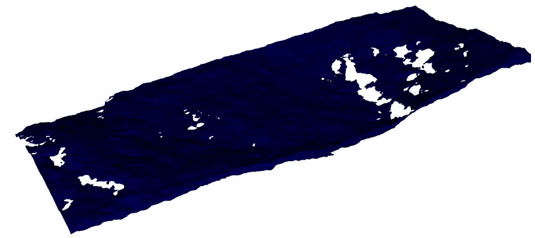
Extended to engineering applications:

- Lithium ion battery electrodes
- Paper manufacturing (hpc4mfg)

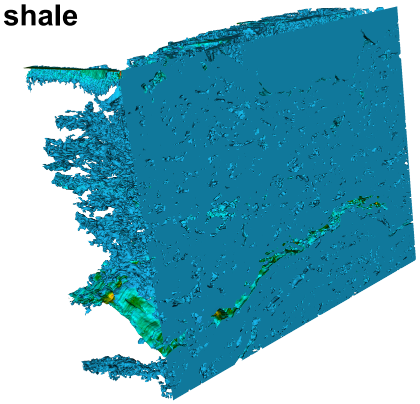
**pH on crushed calcite in capillary tube**



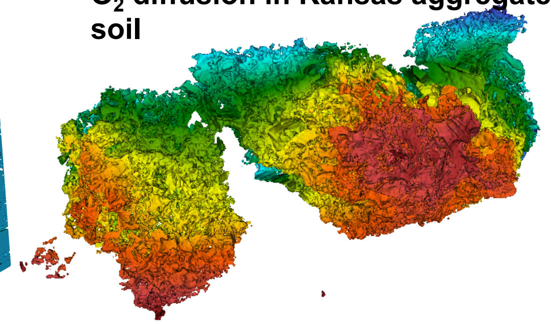
**Transport in fractured**



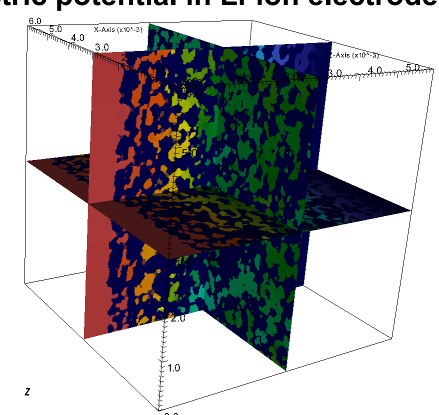
**Flooding in fractured Marcellus shale**



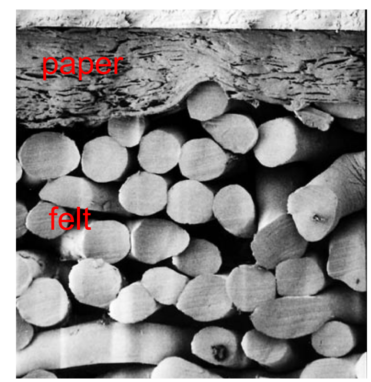
**$O_2$ diffusion in Kansas aggregate soil**



**Electric potential in Li-ion electrode**

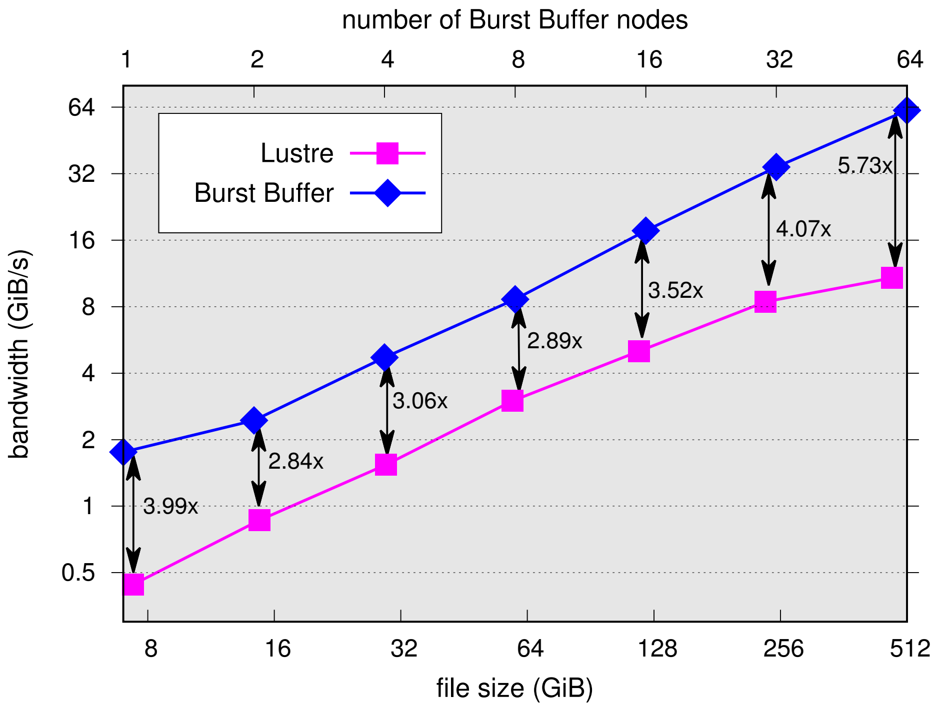


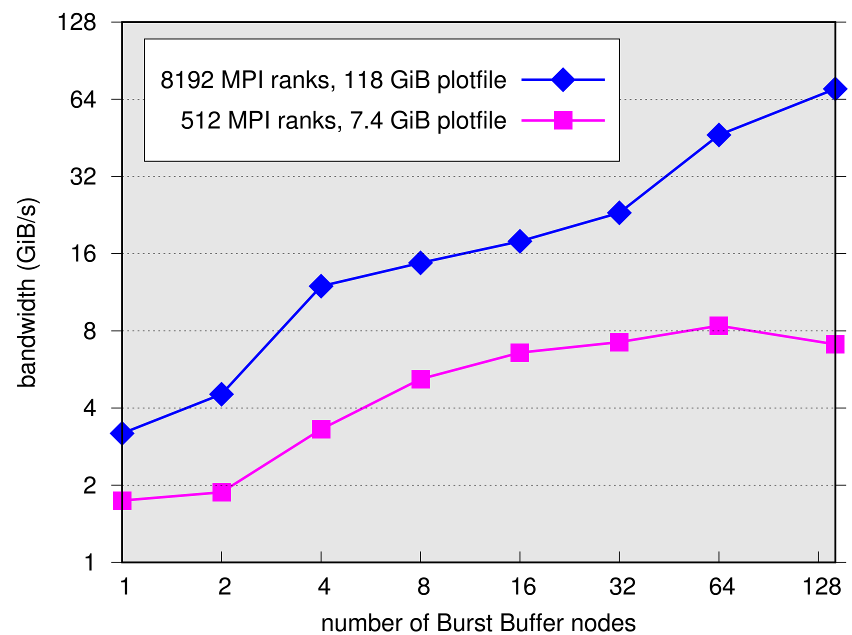
**Paper re-wetting**



# I/O bandwidth study

Collective write to shared file using HDF5 library



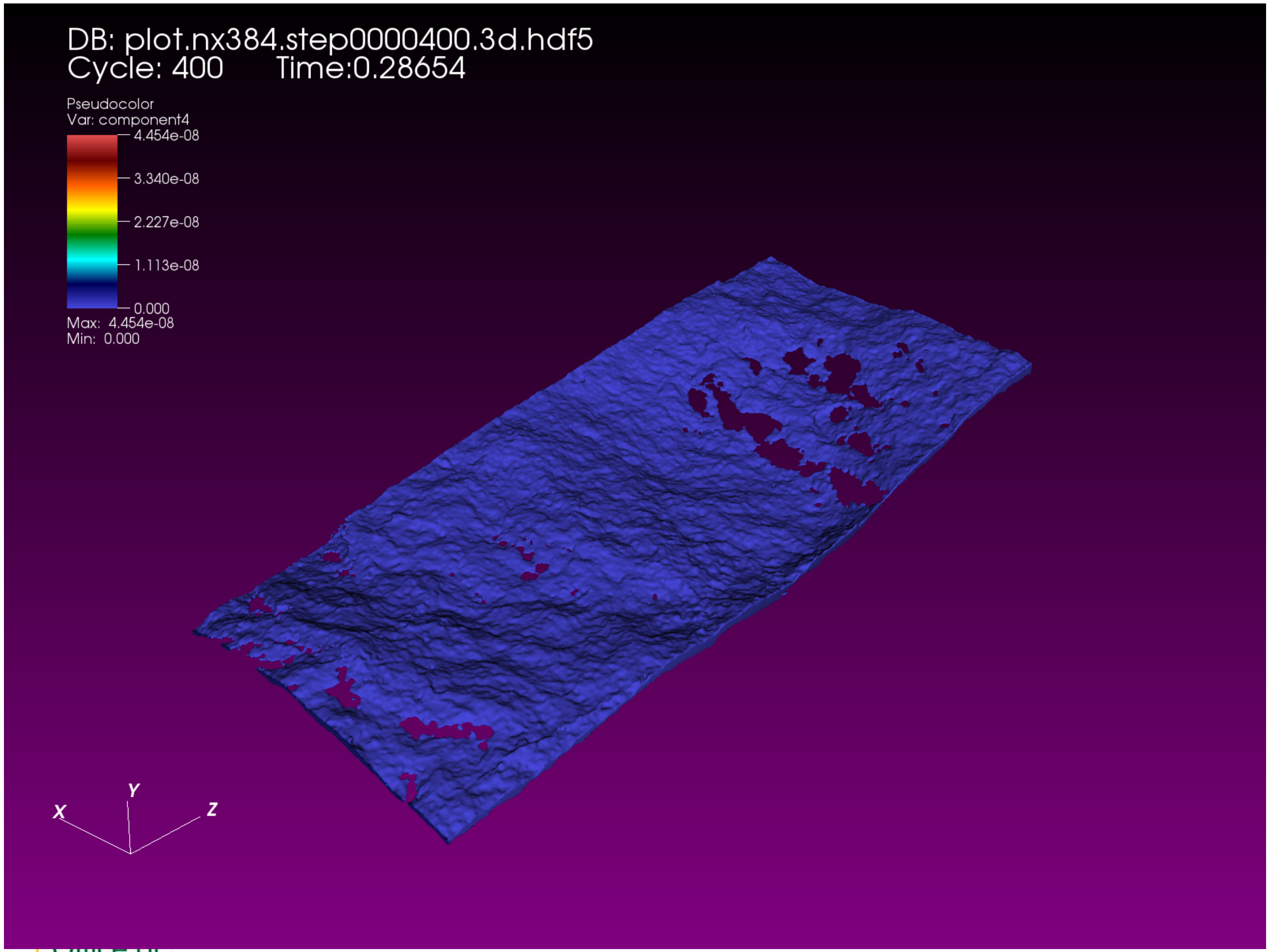




Scaling study for 512 to 32768 MPI tasks for I/O. Number of compute nodes to BB nodes is fixed at 16:1.

Optimal bandwidth study at 2 scenarios.

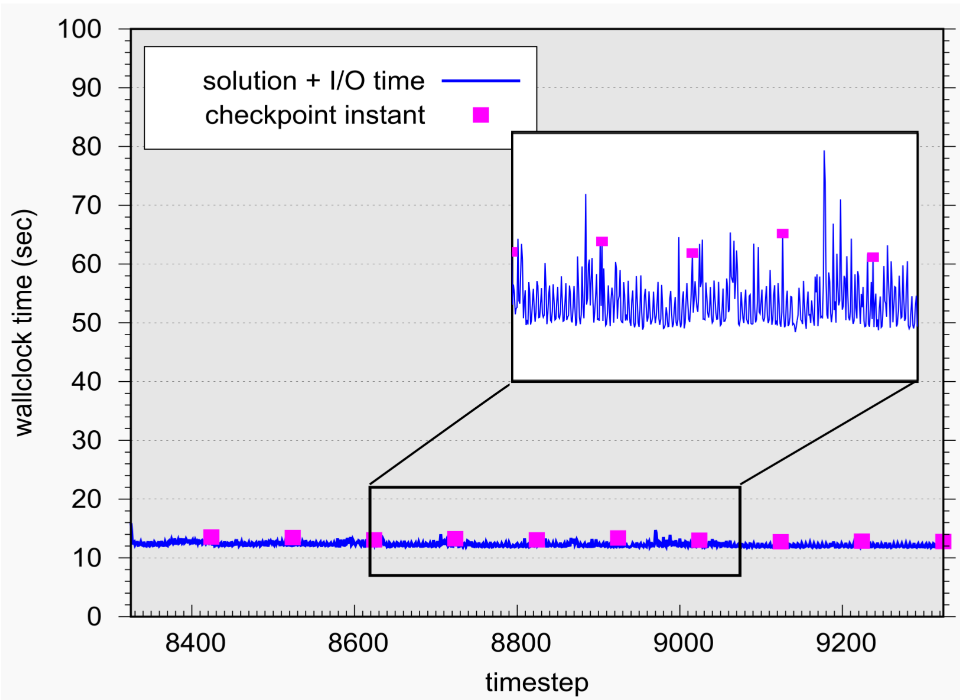
# In-transit visualization: Example 1

**Reactive transport in dolomite**:
Simulation performed on Cori Phase 1: 512 cores used by Chombo-Crunch, 64 cores by VisIt, 144 Burst Buffer nodes for I/O. Plot file size 8GB

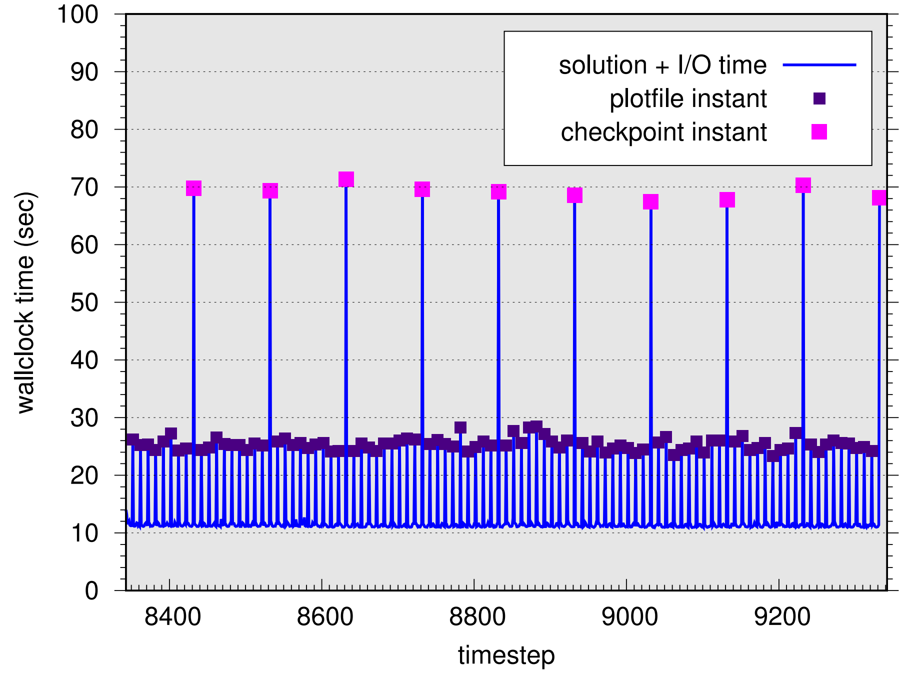


# Wall clock time history

With I/O to Burst Buffer



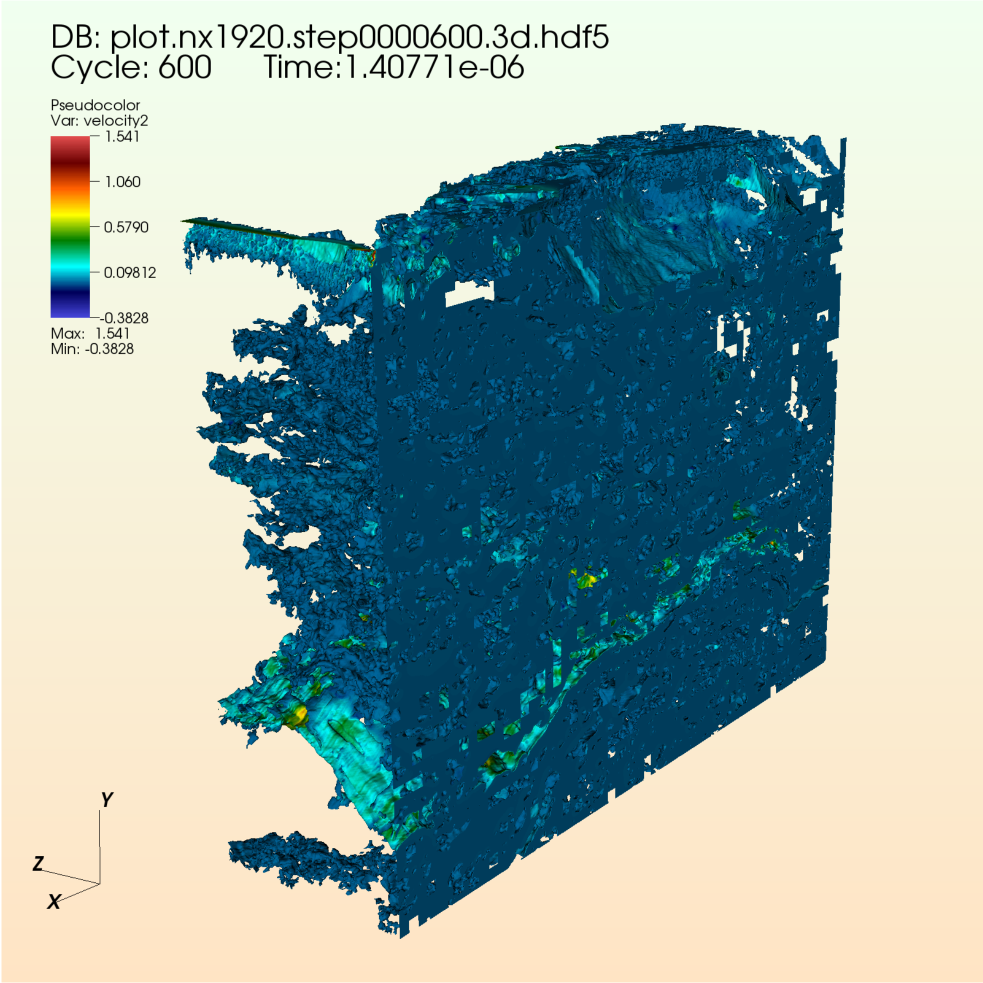

With I/O to Lustre PFS



# In-transit visualization: Example 2

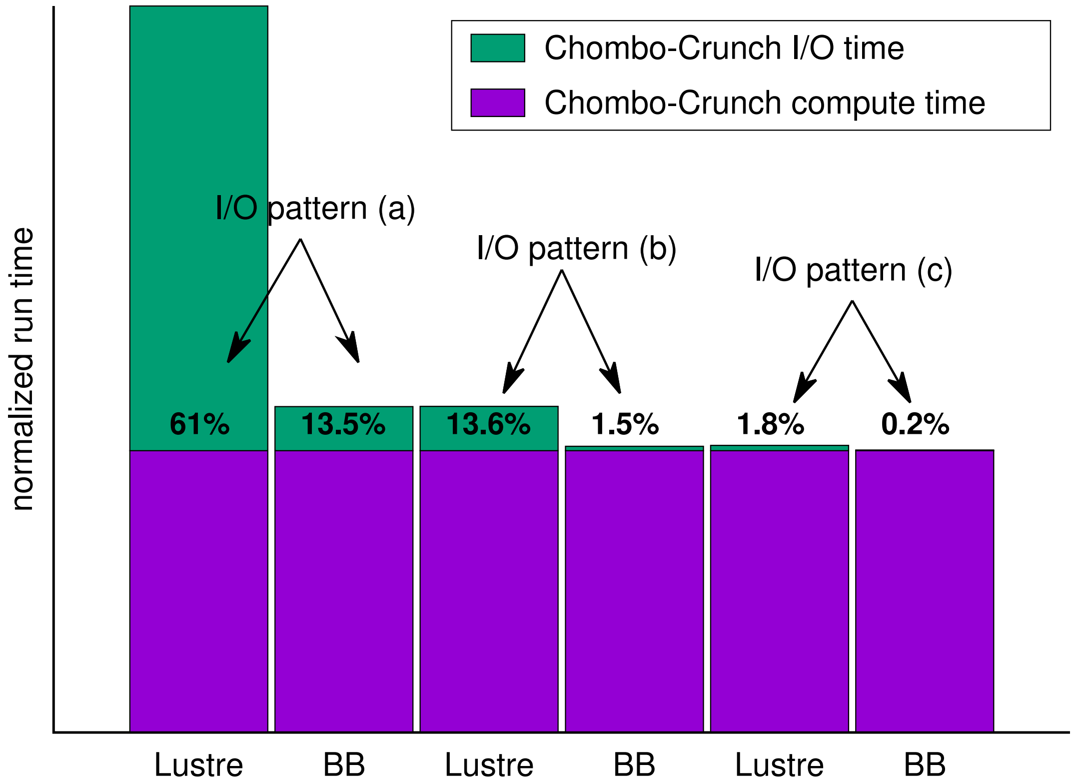
**Reactive transport in shale**

Simulation performed on Cori Phase 1: 32768 cores used by Chombo-Crunch, 512 cores by VisIt, 144 Burst Buffer nodes for I/O. Plot file size 290GB



# Compute time vs I/O time

**(a) High intensity I/O**: write plot file every timestep, checkpointing every 10 timesteps
**(b) Medium intensity I/O**: write plot file every 10 timesteps, checkpointing every 100 timesteps
**(c) Low intensity I/O**: write plot file every 100 timesteps, checkpointing every 500 timesteps



# Summary for 2 benchmarks

| | Shale problem | | Dolomite problem | |
|---|---|---|---|---|
| | I/O to Lustre | I/O to BB | I/O to Lustre | I/O to BB |
| # of timesteps | 670 | | 20000 | |
| plot file size | 288.8 GiB | | 7.46 GiB | |
| checkpoint size | 180 GiB | | 6.12 GiB | |
| Chombo-Crunch compute time per ts | 45.66 s | | 9.87 s | |
| averaged time of writing 1 checkpoint | 136.8 s | 38.4 s | 47.28 s | 1.47 s |
| averaged time of writing 1 plot file | 58.4 s | 3.3 s | 14.45 s | 0.62 s |
| Percentage of Chombo-Crunch I/O: I/O pattern (a) | 61% | 13.5% | 66% | 13.8% |
| Percentage of Chombo-Crunch I/O, I/O pattern (b) | 13.6% | 1.5% | 16.3% | 0.77% |
| Percentage of Chombo-Crunch I/O, I/O pattern (c) | 1.8% | 0.2% | 2.36% | 0.126% |

# Conclusions

- ❑ **In-transit workflow has been proposed and its performance has been assessed on a couple of application examples of Chombo-Crunch subsurface simulation code**
- ❑ **First results show definite I/O improvement and reduction of the overall end-to-end run time**
- ❑ **Utilizing NVRAM memory allows Chombo-Crunch to move to every timestep "postprocessing" while only changing roughly 20 lines of source code in Chombo**
- ❑ **Future work:**
    - Dynamic component load balancing
    - Managing burst buffer capacity
    - Component signaling
    - Including additional components into workflow (e.g. pore graph extractor)

# Burst Buffer Architecture Reality

**BB nodes scattered throughout HSN fabric**
**2 BB blades/chassis (12 nodes/cabinet) in Phase I**

Photo from Glenn Lockwood